MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

ECE4246 – ADVANCED OBJECT-ORIENTED DESIGN WITH JAVA

(All Sections / Groups)

11 OCTOBER 2016
9:00 A.M – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of SIX (6) pages including cover with THREE (3) questions only.
2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.
3. This examination is OPEN-BOOK. Allowed material: Textbook and/or bound notes only.
4. Please print all your answers in the Answer Booklet provided.

Question 1

A student identification (ID) number of University ABC starts with '1', and has 10 digits. The year that a student enrolled into university is indicated by the 2nd and 3rd digits. For example, for enrollment year 2013 a possible ID number is "1131234567". An enrollment year between and inclusive of 2007 and 2016 is valid. Write a program that accepts ID numbers from a user and checks that the ID numbers input are valid according to the format given.

Create a driver class `IDNumbers` which has an instance variable `input` of type `String` for storing input from the user, an `ArrayList` (type `String`) called `IDs` for storing the valid format IDs (starts with '1', and has 10 digits) and another `ArrayList` (type `String`) called `validYear` to indicate whether the ID has a valid year. You may create more variables as necessary.

[5 marks]

Repeatedly prompt for and scan in ID numbers (unless user inputs "NO" to quit).

[4 marks]

Check that an ID entered has valid format and valid year. Store the valid IDs and whether it has a valid year in the `ArrayList`.

[9 marks]

Catch and handle all exceptions that may occur in obtaining user inputs by printing an appropriate message ie. "ID is not numeric: <input>" if user enters non-numeric input. After handling, continue processing more strings from the user.

[2 marks]

Count the number of occurences of valid format ID and valid year. Build and display the output. See Figure Q1.

[5 marks]

```
Enter ID number (NO to quit): 107abc1111
ID is not numeric: 107abc1111
Enter ID number (NO to quit): 1112708888
Enter ID number (NO to quit): 10727088888
Enter ID number (NO to quit): 1061111111
Enter ID number (NO to quit): 1135556666
Enter ID number (NO to quit): 8976
Enter ID number (NO to quit): NO

Result:
1112708888 Valid year
1061111111 Invalid year
1135556666 Valid year
Number of valid format IDs entered: 3
Number of valid year IDs entered: 2
```

Figure Q1: Sample output

Continued...

Question 2

The source code for interface `DrawableItem` is given in Figure Q2a.

```
import java.awt.*;

public interface DrawableItem
{
      public void draw (Graphics g);
      public Dimension getSize ();
}
```

Figure Q2a

Class `IllustrationPanel` is a panel class that has the capability to display a `DrawableItem` object. It has a `DrawableItem` instance variable which is initialized in the constructor. Its method `illustrate()` invokes the draw `method` of the `DrawableItem` instance. The class diagram is given in Figure Q2b.

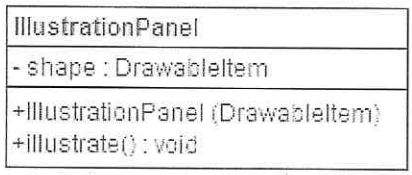

Figure Q2b

NOTE: DO NOT write class `IllustrationPanel`.

Create a class, `Splotch`, which implements `DrawableItem`, and creates a drawing of two overlapping circles (*note: position of circles are not important*). It has instance variables `color` and `size`. Variable `size` is the size of the larger circle and the second circle is half of that. Provide a constructor, accessor and mutator methods, and define the abstract method `draw()`.

[10 marks]

Create an application program, `PopUpSplotch`, which prompts the user to enter a value for size (see Figure Q2c), then creates `Splotch` object of a random colour. Subsequently, instantiate an `IllustratorPanel` object to display the "splotch" in a frame (see Figure Q2d).

[9 marks]

```
Enter size of *splotch*: 100
```

Figure Q2c

Continued...

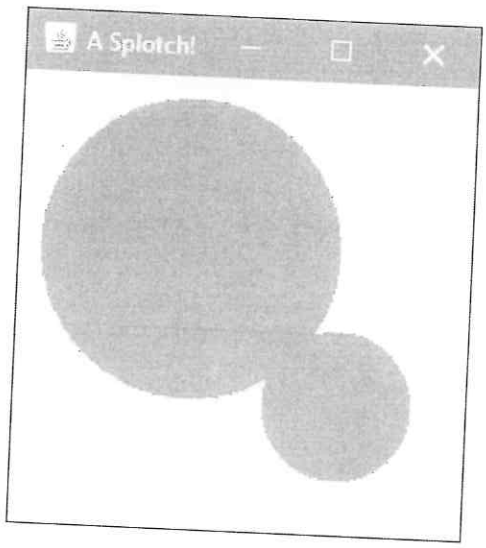

Figure Q2d

Draw the UML class diagrams to show the relationships among **`DrawableItem`**, **`IllustrationPanel`**, **`Splotch`**, **`PopUpSplotch`** and all other classes used in their implementation (if any).

[6 marks]

Question 3

Figure Q3a shows the source code for class `MultiUnit`. This class represents quantities that can be represented by 2 different commonly-used units (eg., distance can be in km or miles).

```
public abstract class MultiUnit
{  protected boolean isDefaultUnit;
   protected double value;

   public MultiUnit (boolean def, double val)
   {  isDefaultUnit = def;
      setValue (val);
   }

   public void setValue (double val)
   {  value = (isValid(val))? val : 0; }

   public double getValue() {     return value;    }

   public void changeForm()
   {  if (isDefaultUnit)
            toAlternateUnit();
      else
            toDefaultUnit();
      isDefaultUnit = !isDefaultUnit;
```


Continued...

```
    }

    public boolean getIsDefault()
    {  return isDefaultUnit;       }

    public String toString()
    {  return Double.toString(value);   }

    public abstract boolean isValid();
    public abstract void toDefaultUnit();
    public abstract void toAlternateUnit();
}
```

Figure Q3a

Create a class **`Weight`**, which is a subclass of **`MultiUnit`**. Class **`Weight`** represents weight either in gram (default unit) or ounce. Define all superclass abstract methods. Method **`isValid`** returns true if the argument is a valid weight quantity and false otherwise. Method **`toDefaultUnit`** converts the value to gram if it is in ounce, and method **`toAlternateUnit`** does the reverse.

[8 marks]

Create class, **`WeightConversionPanel`**, that inherits from **`JPanel`**. It contains a textfield, label and a button. See Figure Q3b.

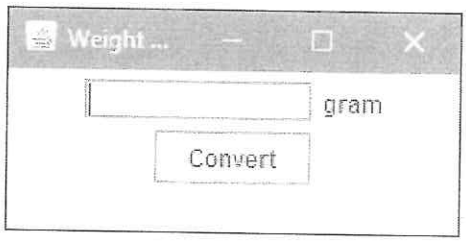

Figure Q3b

Class **`WeightConversionPanel`** uses instances of **`Weight`** to allow the conversion of weight value from gram to ounce. The button "Convert" changes the value in the textfield in its current unit (either gram or ounce) to the other one. It also changes the label to reflect the conversion.

[13 marks]

Sample output: For input shown in Figure Q3c, the result of clicking "Convert" is shown in Figure Q3d. If "Convert" is clicked again for the value displayed in Figure Q3d, the result reverts back to the original value (Figure Q3c).

Continued...

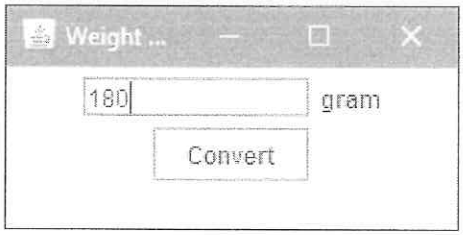

Figure Q3c: Original Inputs

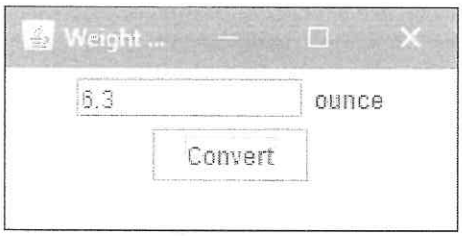

Figure Q3d: After clicking "Convert"

NOTE: DO NOT write the application / driver program.

State **TWO (2)** design patterns used in `MultiUnit`, `Weight`, and `WeightConversionPanel`, and briefly explain how they apply in the classes.

[4 marks]

Hint:
$1\ ounce = 28.35\ gram$

End of Paper